

Consiglio Nazionale delle Ricerche

Clustering Transactional Data

Fosca Giannotti, Cristian Gozzi, Giuseppe Manco

Technical Report
CNUCE-B4-2001-004

CNUCE

Pisa



Clustering Transactional Data

Fosca Giannotti, Cristian Gozzi, and Giuseppe Manco

CNUCE-CNR, Pisa Research Area
Via Alfieri 1, 56010 Ghezzano (PI), Italy
email: {F.Giannotti,G.Manco}@cnuce.cnr.it, G.Gozzi@guest.cnuce.cnr.it

Abstract. In this paper we present a partitioning method which crosses the limitations of the traditional approaches to clustering of transactional data. We present a modification of the standard K-Means algorithm, which has a good scalability on the number of objects and attributes, but can only work with numeric vectors of fixed length. Our algorithm works with transactions, i.e., sets of variable size containing discrete-valued attributes. We adapt the standard definition of mathematical distance used in the K-Means algorithm to represent transactions dissimilarity, and redefine the notion of cluster center. In our algorithm a cluster center is a typical transaction representing the common properties of cluster elements. The resulting algorithm maintains the formal complexity of standard K-Means, but substantially outperforms the latter on both synthetic and real data. Experiments with such datasets show the good scalability of our algorithm, in term of both the size of the dataset and the average size of the transactions.

1 Introduction

1.1 Motivations

Clustering is a useful technique for grouping data in partitions, called clusters, based on values of their attributes. Contributing areas of research include data mining, statistics, machine learning, pattern recognition. The following are typical requirements of clustering algorithms in data mining:

- **Scalability:** a large database can contain millions of objects. Clustering on a sample may be ineffective, so we need algorithms that can efficiently work with the entire database.
- **High dimensionality:** many clustering methods are good at handling low-dimensional data, but real datasets may have many dimensions. It is challenging to cluster objects in high dimensional spaces, even in consideration that data can be very sparse and highly skewed.
- **Ability to deal with different types of attributes:** many algorithms are designed to handle only attributes with numeric values. However, real data can be of boolean, categorical, textual type.
- **Interpretability and usability of results:** data mining applications require clusters description that can be easily interpreted by final user. For this reason it's necessary to represent clusters using a simple formalism.

In such context, the problem of clustering transactional data, i.e., tuples of variable size, requires all the above capabilities:

- in most application domains transactions can be very huge: for example, daily web user sessions of a small web community (e.g., a university) are almost hundreds of thousands of order of magnitude.
- the number of elements that can appear in a transaction is usually huge. An example is the number of web pages that can be visited in a user session.
- A transaction can contain both discrete-valued and continuous-valued data.
- each transactional cluster should have a cluster representative: a typical transaction subsuming all the transactions belonging to the cluster.

The need to develop algorithms for transactional clustering comes from their various relevant applications in real-life problems. The primary motivation of our work was web data clustering. Log data typically represents sets of single accesses of web users to web resources. The set of one user accesses can be split into *sessions*. A session is a set of web pages visited by a user in a "semantically homogeneous" way (e.g., in order to find useful information about user interests, or to improve proxy cache performances). A problem strictly connected to the previous is clustering of purchase transactions, where a market basket correspond to a session and the purchased items to single accesses.

Efficient clustering of transactions is also a requirement in text mining [10, 17, 21]. The main idea here is to cluster web documents according to the keywords they contains. Web documents can be pre-processed and represented by a set of significant words. Hence, a transactional clustering algorithm can help finding homogeneous groups of documents, according to the keywords they contain. To each cluster we can then associate a representative (a set of words) that can be used to label the documents belonging to the cluster. Another potential application of transactional clustering is the clustering of sparse datasets, i.e., datasets whose attributes contain a large number of null values.

The traditional clustering approaches to these kinds of applications, however, exhibit some drawbacks. In the current literature, transactional clustering is usually dealt with hierarchical methods. Hierarchical clustering is often presented as the better quality clustering approach, but it is rather inefficient on large datasets for its quadratic complexity. Moreover, it is difficult to generate a representative value providing an easy interpretation of the clusters population. In contrast, classical partitioning methods, like K-Means and its common variants, have a linear scalability on the number of objects and attributes, and consequently are efficient on large databases.

However, these methods do not provide an adequate representation formalism for tuples of variable size containing categorical data. The most suited approach to cluster categorical data with K-Means based algorithms was presented in [15]. In this method every single categorical attribute is converted into a set of boolean attributes. The cardinality of a set is the number of distinct values that the categorical attribute can assume and every boolean attribute takes the value 1 if the corresponding categorical value is in the object, 0 otherwise. As a result, even a small set of items gives raise to a curse of high dimensionality.

Furthermore, the traditional concepts of similarity and center of a cluster used by standard K-Means are not significant for transactional data. The euclidean distance, in fact, does not represent the real semantic distance between two transactions, and the notion of cluster mean is not adequate to represent the cluster characteristics.

1.2 Objectives

The objective of this paper is to implement a partitioning method for transactional clustering which crosses the limitations of traditional K-Means algorithms. We develop a method that works on sets with variable cardinality containing discrete values, and uses viable concepts of mathematical distance and cluster center. The main idea is that of re-defining a cluster center as a transaction representing the common properties of cluster elements. Similarity inside a cluster is hence measured by using the cluster representative, that becomes also a natural tool for finding an explanation of the cluster population.

The plan of the paper is as follows. In section 2 we provide an overview of the traditional approaches to transactional clustering. Section 3 provides the formal foundations of the clustering algorithm that is shown in section 4. Finally, in section 5 we show formal and empirical results to prove that the algorithm is both efficient and effective on transactional data.

2 Related works

The problem of transactional clustering is related to two classes of problems: clustering of categorical attributes and clustering of variable length sets. An example of algorithm that deals with such problems in a unified way was introduced by Guha et al. [8]. Here, the ROCK algorithm is presented, an agglomerative hierarchical method for clustering sets of categorical values. A given dataset is partitioned in an

agglomerative way, using the number of common neighbors as a criterion for merging the subclusters. A neighbor is a set with a similarity value greater than a fixed threshold. The similarity measure is the Jaccard index. ROCK complexity is quadratic w.r.t. the number of input objects. To overcome this limitation, the authors define a two-steps approach: at the first step, a sample of the dataset is clustered, in order to obtain a suitable set of clusters. In the next step the entire dataset is considered, and each tuple is assigned to one of the clusters obtained from the previous step.

An alternative to the previous method is that of exploiting partitioning methods such as, e.g., K -Means and its variants, which have a linear scalability on the size of the dataset. The K -Modes algorithm, proposed by Huang [11], is an extension of the standard K -Means method for categorical domains. An approach similar to the former is described by Gupta et al in [9]. The algorithm is defined to work with vectors of categorical attributes and defines as cluster center the vector containing the modes of the categorical attributes. The main drawback of this method is the necessity of dealing with boolean representations of tuples with variable size, thus making the approach uneffective when dealing with sets having a large number of items.

Another method for clustering objects with categorical attributes, called CACTUS, was proposed by Ganti et al. [5]. The algorithm has three phases: summarization, clustering and validation. In the first phase the dataset is modeled using summaries. Inter-attribute summaries represent strictly connected pairs of values belonging to distinct attributes. Intra-attribute summaries describe similarities between values of the same attribute. The results of the summarization phase are used for the computation of a set of candidate clusters. In the last phase real clusters are obtained from candidate clusters. The summarization phase need complex computations. The cost of such phase is acceptable provided that the cardinality of the domains of is small. To extend CACTUS in order to handle large attribute value domains the authors suggest to change original domain with an abstraction having lower cardinality.

Other approaches focus on the problem of clustering transactional data. Wang and others [20] describe a clustering method for transactions based on the paradigm of allocation and refinement used in partitional clustering algorithms, but with important differences. Clusters are formed iteratively using the concept of large item, that is an item frequently occurring in transactions and minimizing the total cost (defined in terms of both intra-cluster and inter-cluster similarity).

Strehl and Ghosh present OPOSSUM [18], a similarity-based clustering method particularly attuned to market baskets. The algorithm is based on constrained, weighted graph partitioning. The Computational complexity of the algorithm, however, is $O(N^2M)$, where M is the cardinality of distinct items. Cadez, Gaffney and Smyth [1] present a probabilistic framework for clustering objects when data measurements are not vectors of fixed dimensionality. Each cluster is represented by a finite state Markov model, and every object has probability P of belonging to cluster K . The authors propose a variant of the EM (Expectation-Maximization) algorithm. The main idea is that each transaction is a sequence of events performed by an individual, and can be modeled using Markov chains. Although the approach is attractive, the generation of each cluster requires the construction of an $M \times M$ transition matrix (where M is the number of distinct values the sequences can contain).

3 Problem statement

A clustering problem can be stated in many different ways. We shall refer to the following formalization, that views clustering mainly as a partitioning of objects. Given a dataset $D = \{x_1, \dots, x_n\}$ of objects, a clustering is a partitioning of D into subsets, where each cluster represent one of such subsets. Let w_{ij} be a weight representing the similarity between the objects x_i and x_j and y_{ij} a variable such that

$$y_{ij} = \begin{cases} 1 & \text{if } x_i \text{ and } x_j \text{ belong to the same cluster} \\ 0 & \text{otherwise} \end{cases}$$

The clustering problem can hence be stated as the problem of finding the assignments to y_{ij} , $1 \leq i \leq j \leq n$ that satisfy the following constraints:

$$\text{maximize} \quad \sum_{1 \leq i < j \leq n} w_{ij} y_{ij}$$

$$\begin{aligned}
\text{provided that } & y_{ij} = y_{ji} \\
& y_{ij} + y_{jk} - y_{ik} \leq 1 \\
& y_{ij} - y_{jk} + y_{ik} \leq 1 \\
& -y_{ij} + y_{jk} + y_{ik} \leq 1 \\
& 1 \leq i < j < k \leq n
\end{aligned}$$

Unfortunately this problem is NP-hard [7], and hence we have to define heuristics methods in order to find a suitable grouping.

The two main approaches to clustering are hierarchical and partitional clustering [3, 4]. Hierarchical methods are cited in the literature as the better quality clustering approaches, but they are limited because of their quadratic complexity over the number of object under consideration. In this paper we focus on partitional algorithms that have linear complexity. The most well-known partitional approach is the K -Means algorithm, introduced by MacQueen [12]. For a given parameter K , K -Means partitions D into K clusters that guarantee an high intra-cluster similarity and a low inter-cluster similarity. Each object x_i is assigned to a cluster j according to its distance $d(x_i, m_j)$ from a value m_j representing the cluster itself. m_j is called the *center* (or *representative*) of the cluster.

Definition 1. Given a set of objects $D = \{x_1, \dots, x_n\}$, the K -Means algorithm finds a partition $\mathcal{C} = \{C_1 \dots C_k\}$, of D such that:

1. each C_i is associated to a center m_i
2. $x_i \in C_j$ if and only if $d(x_i, m_j) < d(x_i, m_l)$ for $1 \leq l \leq k, j \neq l$
3. The partition \mathcal{C} minimizes $\sum_{i=1}^k \sum_{x_j \in C_i} d(x_j, m_i)$

□

The algorithm works as follows. First of all, K objects are randomly selected from D . Such objects correspond to some initial cluster centers, and each remaining object in D is assigned to the cluster satisfying condition 2. Next, the algorithm iteratively recomputes the center of each cluster and re-assigns each object to the cluster of the nearer center. The algorithm terminates when the centers do not change anymore. In that case, in fact, condition 3 holds.

The general schema of K -Means, shown in fig. 1, is parametric to the functions d and rep , that formalize the concepts of distance and center. Such concepts in turn are parametric to the domain of D .

Definition 2. Given a domain \mathcal{U} equipped with a distance function $d : \mathcal{U} \times \mathcal{U} \mapsto \mathbb{R}$ and a set $\mathcal{S} = \{x_1, \dots, x_m\} \subseteq \mathcal{U}$, the center of \mathcal{S} is the element that minimizes the sum of the distances:

$$rep(\mathcal{S}) = \min_{v \in \mathcal{U}} \sum_i d(x_i, v)$$

□

Example 3. When $\mathcal{U} = \mathbb{R}^n$, a standard definition of d is the *Minkowski* distance:

$$d(x, y) = \left(\sum_{i=1}^n (x_i - y_i)^p \right)^{1/p}$$

where p is a prime integer. When $p = 2$, the distance d is referred to as the euclidean distance, and the center of a set $\mathcal{S} = \{x_1, \dots, x_m\}$ is computed as

$$rep(\mathcal{S}) = \frac{1}{m} \sum_i x_i$$

□

Algorithm K -Means(D, K);

Input : a dataset $D = \{x_1, \dots, x_N\}$ of objects, the desired number K of clusters.

Output : a partition $\mathcal{C} = \{C_1, \dots, C_k\}$ of D in K clusters.

Method :

- Randomly choose x_{i_1}, \dots, x_{i_k} and set $m_j = x_{i_j}$ for $1 \leq j \leq k$.
 - Repeat
 - for each j , set $C_j = \{x_i | d(x_i, m_j) < d(x_i, m_l), 1 \leq l \leq k\}$;
 - set $m_j = \text{rep}(C_j)$ for $1 \leq j \leq k$;
- until m_j do not change.
-

Fig. 1. The K -Means algorithm

Example 4. Let us consider $\mathcal{U} = \{v_1, \dots, v_n\}$. The mismatch-count distance is defined as follows [11]:

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

It can be shown that, for a given set $S = \{x_1, \dots, x_m\} \subseteq \mathcal{U}$, the center is given by the mode of the set:

$$\text{rep}(S) = v \in \mathcal{U} \text{ such that } \text{freq}(v, S) > \text{freq}(v_i, S), \text{ for } 1 \leq i \leq n$$

The approach can be easily generalized to the multidimensional case $\mathcal{U} = \mathcal{U}_1 \times \mathcal{U}_2 \times \dots \times \mathcal{U}_n$, where $\mathcal{U}_i = \{v_{i1}, \dots, v_{in_i}\}$. In such case the distance is defined as

$$d(x, y) = \sum_{i=1}^n \delta(x_i, y_i)$$

and the representative is given by the vector containing the mode of each attribute. □

Transactional data, in this paper, is referred to as vectors of variable size, containing categorical values.

Definition 5. Given a set $\mathcal{I} = \{a_1 \dots a_m\}$, where a_i is a categorical value (henceforth called *item*), The domain of transactional data is defined as

$$\mathcal{U} = \text{Powerset}(\mathcal{I})$$

A subset $I \subseteq \mathcal{I}$ is called an *itemset*. □

Example 6. Let us suppose that a_i represents a web URL, and that \mathcal{I} models all the pages contained within a web server. An itemset can then represent a typical web user session within the web server, i.e., the set of pages a user has accessed within the given web server.

3.1 Dissimilarity measure

As we already mentioned, the standard approach used to deal with transactional data in clustering algorithms is that of representing such data by means of fixed-length boolean attributes.

Example 7. Let us suppose $\mathcal{I} = \{a_1, \dots, a_{10}\}$. We can represent the itemsets $I_1 = \{a_1, a_2\}$, $I_2 = \{a_1, a_2, a_3, a_5, a_6, a_7\}$ and $I_3 = \{a_4\}$ by means of the following vectors:

$$\begin{array}{l} I_1 \quad [1, 1, 0, 0, 0, 0, 0, 0, 0, 0] \\ I_2 \quad [1, 1, 1, 0, 1, 1, 1, 0, 0, 0] \\ I_3 \quad [0, 0, 0, 1, 0, 0, 0, 0, 0, 0] \end{array}$$

In this representation, position i of the boolean vector represents object a_i . A value 1 corresponds to the presence of a_i to the transaction, while a value 0 corresponds to its absence. □

In principle, the above representation allows to directly apply the concepts of distance and center defined in the previous section. Such approaches, however, do not capture our intuitive idea of transaction similarity. In the above example, I_2 is more similar to I_1 than to I_3 , since there is a partial match between the transaction.

Example 8. Using the euclidean distance, we obtain $d(I_1, I_2) = 2$ and $d(I_1, I_3) = \sqrt{3}$; thereby, I_1 is more similar to I_3 than to I_2 . These result is clearly in contrast with our intuition, since the only property that I_1 and I_3 have in common is the absence of most of the items. \square

Example 9. Using the mismatch-count distance, we obtain $d(I_1, I_2) = 4$ and $d(I_1, I_3) = 3$. Again, this result does not capture our intuitive notion of similarity between transactions. \square

The problem with the above dissimilarity measures is in the fact that they consider both the presence and the absence of an item within a transaction. As a consequence, sparse transactions (i.e., transactions containing a very small subset of \mathcal{I}) are very likely to be similar, even if the items they contain are quite different. A solution to this problem is given by the *Jaccard Coefficient* [14, 19]. This measure is based on the idea that similarity between objects, represented by boolean vectors, is directly proportional to the number of common values, and inversely proportional to the number of different values for the same attribute.

Definition 10. Given $x, y \in \mathbb{R}^n$, the Jaccard coefficient is defined as

$$s(x, y) = \frac{N_{11}}{N_{11} + N_{10} + N_{01}}$$

where $N_{11} = \sum_{i=1}^n x_i y_i$, $N_{10} = \sum_{i=1}^n x_i (1 - y_i)$ and $N_{01} = \sum_{i=1}^n (1 - x_i) y_i$. \square

A distance measure can be straightforwardly defined from the Jaccard coefficient. For example, we can define $d(x, y) = 1 - s(x, y)$.

Example 11. Let us consider again the transactions of example 7. Using the above distance measure, we obtain $d(I_1, I_2) = 2/3$ and $d(I_1, I_3) = 1$. \square

A more intuitive but equivalent way of defining the above distance function can be provided. Given two itemsets I and J , we can represent $d(I, J)$ as the (normalized) difference between the cardinality of their union and the cardinality of their intersection:

$$d_J(I, J) = 1 - \frac{|I \cap J|}{|I \cup J|}$$

Lemma 12. $d_J(I, J)$ is a mathematical distance.

Proof. See [6] \square

3.2 Cluster representative

Another problem to face with is a suitable definition of the cluster representative for transactional data. In definition 2 we have specified the criteria for obtaining the cluster representative once the distance function is fixed. Intuitively, a cluster representative for transactional data should model the content of a cluster, in terms, e.g., of the elements that are most likely to appear in a transaction belonging to the cluster. In such case, in fact, the interpretation of a cluster from its representative should be trivial. The euclidean and mismatch-count distances provide an interesting approximation of such a desired behavior of the cluster representative.

Example 13. Let us consider again the transactions of example 7. The euclidean distance over the vectorial representation of the transactions provides a cluster representative enclosing the frequency distributions of each item within the cluster.

$$rep(\{I_1, I_2, I_3\}) = [2/3, 2/3, 1/3, 1/3, 1/3, 1/3, 1/3, 0, 0, 0]$$

Such a model provides a full representation of all the features within the cluster. However, the representative is not a transaction by itself. Hence, as long as the K -Means algorithm recomputes cluster representatives, the intra-cluster similarity can loose its originary significance. \square

Example 14. The Mismatch-count distance produces the following representative for the transactions of example 7:

$$rep(\{I_1, I_2, I_3\}) = \{a_1, a_2\}$$

Here, items a_1 and a_2 are the most frequent items actually appearing within the cluster. A problem with such an approach is that less frequent items, i.e., items appearing in many but not in most of the transactions, are not included in the cluster representative. \square

The use of the Jaccard distance makes computing a cluster representative problematic. In fact, the problem of minimizing the expression $\sum_i d_J(x_i, c)$ given a set $\{x_1, \dots, x_m\}$ cannot be solved in polynomial time if c is required to be an itemset [7]. Moreover, the cluster representative can have more than one cluster center, as the following example shows.

Example 15. Consider the transactions $T_1 = \{a_1, a_2\}$, $T_2 = \{a_1, a_3\}$ and $T_3 = \{a_1, a_4\}$. Each of them is a good candidate as a cluster representative, since it minimizes the sum of the distances:

$$\begin{aligned} \sum_{i=1}^3 d_J(T_1, T_i) &= 4/3 & \sum_{i=1}^3 d_J(T_2, T_i) &= 4/3 & \sum_{i=1}^3 d_J(T_3, T_i) &= 4/3 \\ \sum_{i=1}^3 d_J(\{a_1\}, T_i) &= 3/2 & \sum_{i=1}^3 d_J(\{a_1, a_2, a_3\}, T_i) &= 17/12 & \sum_{i=1}^3 d_J(\{a_1, a_2, a_4\}, T_i) &= 17/12 \\ \sum_{i=1}^3 d_J(\{a_2, a_3, a_4\}, T_i) &= 17/12 & \sum_{i=1}^3 d_J(\{a_1, a_2, a_3, a_4\}, T_i) &= 3/2 & & \end{aligned}$$

\square

In order to overcome such problems, we can compute an approximation that resembles the cluster representatives associated to the euclidean and mismatch-count distances. Union and intersection seem good candidates to start with.

Lemma 16. Given a set $S = \{x_1, \dots, x_m\}$, $rep(S) \subseteq \bigcup_i x_i$

Proof. Let $U = \bigcup x_i = \{a_1 \dots a_n\}$ and $R = rep(S)$. Let us suppose that $R \not\subseteq U$. Hence, $R = S_1 \cup S_2$ such that:

1. $S_1 \subseteq U$
2. $S_2 \cap U = \emptyset$
3. $|S_2| = w > 0$

Suppose, for a given x_i that $|S_1 \cup x_i| = k_i$ and $|S_1 \cap x_i| = h_i$. Hence,

$$\begin{aligned} & d_J(x_i, S_1) \\ = & \{ \text{definition of } d_J \} \\ & (k - h)/k \\ < & \{ \text{trivial arithmetic} \} \\ & (k + w - h)/(k + w) \\ = & \{ \text{definition of } d_J \} \\ & d_J(x_i, S_1 \cup S_2) \end{aligned}$$

Now, since $S_1 \subseteq U$, there is at least one element x_j such that $h_j > 0$. As a consequence,

$$\sum_i d_J(x_i, S_1) < \sum_i d_J(x_i, R)$$

that is clearly a contradiction, by definition of R . \square

The above lemma shows that the only elements we need to consider in order to compute the representative are contained in the union of the elements of the cluster. The following example shows, however, that the cluster center can be different from the union.

Example 17. Let us consider a cluster containing the elements $I_1 = \{a_1, a_2, a_3\}$, $I_2 = \{a_1, a_3, a_4\}$ and $I_3 = \{a_1, a_2, a_5\}$. Their union is hence $U = \{a_1, a_2, a_3, a_4, a_5\}$. Now $\sum_{j=1}^3 d_J(I_j, U) = 9/5$. However, for the set $C = \{a_1, a_2, a_3\}$, we have $\sum_{j=1}^3 d_J(I_j, C) = 1$. \square

The idea of approximating the cluster center with the union has some drawbacks that make such choice unpractical. First of all, the resulting representative can have a huge cardinality because of the heterogeneity of objects. If transactions contain a large number of distinct values and the cluster contains a reasonable amount of transactions, the probability their union has a large cardinality is very high. The second problem is represented by the fact that the union contains all the values in the objects without considering their frequencies. In this case, in fact, the resulting intra-cluster similarity can be misleading, since the cluster may contain elements with little elements in common.

To overcome the above problems, one can think to use the intersection of the transactions as an approximation of the representative. And, indeed, the intersection is actually contained in a cluster representative.

Lemma 18. Given a set $\mathcal{S} = \{x_1, \dots, x_m\}$, $\bigcap_i x_i \subseteq rep(\mathcal{S})$

Proof. Again, we prove the assertion by contradiction. Let $I = \bigcap_i x_i = \{a_1, \dots, a_k\}$, and $R = rep(\mathcal{S})$. Let us suppose that $I \not\subseteq R$. Hence,

1. $R = S_1 \cup S_2$
2. $S_1 \subset I$
3. $S_2 \cap I = \emptyset$.

Now we choose $J = I \cup S_2$ and show that, for each x_i , $d_J(x_i, J) < d_J(x_i, R)$.

$$\begin{aligned} & d_J(x_i, I \cup S_2) \\ = & \frac{\{ \text{definition of } d_J \}}{|x_i \cup I \cup S_2| - |x_i \cap (I \cup S_2)|} \\ = & \frac{\{ \text{intersection distributes over union} \}}{|x_i \cup S_2| - |(x_i \cap I) \cup (x_i \cap S_2)|} \\ = & \frac{\{ I \subseteq x_i \text{ and } I \cap (x_i \cap S_2) = \emptyset \}}{|x_i \cup S_2| - |I| - |x_i \cap S_2|} \\ < & \frac{\{ S_1 \subset I \}}{|x_i \cup S_2| - |S_1| - |x_i \cap S_2|} \\ = & \frac{\{ S_1 \subseteq x_i \text{ and } S_1 \cap (x_i \cap S_2) = \emptyset \}}{|x_i \cup S_2| - |(x_i \cap S_1) \cup (x_i \cap S_2)|} \\ = & \frac{\{ \text{intersection distributes over union} \}}{|x_i \cup S_1 \cup S_2| - |x_i \cap (S_1 \cup S_2)|} \\ = & \frac{\{ \text{definition of } d_J \}}{d_J(x_i, R)} \end{aligned}$$

$$T_1 = \{1, 6, 9, 4\} \quad T_2 = \{3, 5, 9, 4\} \quad T_3 = \{0, 8, 4, 7, 2\} \quad T_4 = \{5, 7, 9\}$$

$$T_5 = \{9, 7, 5\} \quad T_6 = \{1, 3, 2\} \quad T_7 = \{0, 8, 9, 7, 5\} \quad T_8 = \{3, 4, 5, 8\}$$

The optimal center is $\{3, 4, 5, 7, 8, 9\}$, containing all items with frequency $\geq 3/8$.

$$T_1 = \{1, 2, 3, 4\} \quad T_2 = \{5, 6, 7, 8, 9\} \quad T_3 = \{1, 2, 10, 11\} \quad T_4 = \{12, 13, 14\}$$

$$T_5 = \{5, 15, 16\} \quad T_6 = \{3, 11, 17, 18\} \quad T_7 = \{1, 6, 19, 20, 21\}$$

The optimal center is $\{1, 2, 3, 5, 6, 9, 11\}$, containing all items with frequency $\geq 2/7$.

Fig. 2. Examples of optimal cluster centers.

On the basis of the above result, we obtain that $\sum_i d_J(x_i, J) < \sum_i d_J(x_i, R)$, which is clearly a contradiction. \square

Also in this case, the intersection does not necessarily correspond to the cluster representative.

Example 19. Let us consider again the transactions of example 17. The intersection of such transactions is $I = \{a_1\}$, which has the sum of distances $\sum_{j=1}^3 d_J(I_j, I) = 2$. Again, such a value is greater than the value associated to the set $C = \{a_1, a_2, a_3\}$. \square

Again, it can be unpractical to approximate the cluster representative with the intersection. With clusters densely populated, it is very likely to obtain an empty intersection. On the other side, the cluster representative cannot be empty, as the following result shows.

Lemma 20. *Given a set $S = \{x_1 \dots x_m\}$, $rep(S)$ is not empty.*

Proof. We proceed by contradiction. Assume that $rep(S) = \emptyset$. It is easy to see that $\sum_i d_J(x_i, \emptyset) = m$. Let us consider $a \in x_j$, for some given j . We can split S into two subsets: $D = \{x_i : a \in x_i\}$ and $E = \{x_i : a \notin x_i\}$. Clearly, $D \neq \emptyset$. As a result, we obtain:

$$\begin{aligned} & \sum_{i=1}^m d_J(x_i, \{a\}) \\ &= \{ \text{definition of } \sum \} \\ &= \sum_{i \in D} d_J(x_i, \{a\}) + \sum_{i \in E} d_J(x_i, \{a\}) \\ &= \{ \text{definition of } d_J \} \\ &= \sum_{i \in D} (|o_i| - 1) / |o_i| + \sum_{i \in E} 1 \\ &< m \end{aligned}$$

that is clearly a contradiction. \square

A property of cluster representatives is that frequent items are very likely to belong to them. Figure 2 shows two examples of this behavior. Clearly, such a property is not true in general. Consider, for example, the transactions $T_1 = \{a_1, a_3, a_4\}$, $T_2 = \{a_1, a_5, a_6\}$, $T_3 = \{a_1, a_7, a_8\}$, $T_4 = \{a_2\}$. The representative of the above transactions is $\{a_2\}$, but the most frequent item is a_1 .

The main problem with the above example is in the heterogeneity of the transactions. In homogeneous clusters, such a drawback can hardly hold. In each case, a simple way of controlling such an effect can be that of specify the desired degree of frequency-based similarity. We introduce a user-defined threshold value γ over the frequency of the items appearing in the union. Such a parameter represents the intra-cluster similarity degree desired by the user, and corresponds to the minimum percentage of occurrences an item must have to be inserted into the approximation of the cluster representative.

Definition 21. For a given set $S = \{x_1, \dots, x_m\}$. Let $\gamma \in [0, 1]$. The representative of S is defined as

$$rep(S) = \left\{ v \in \bigcup_i x_i \mid freq(v, S) / m \geq \gamma \right\}$$

where $freq(v, S) = |\{x_i \mid v \in S\}|$. \square

Greater γ values correspond to a stronger intra-cluster similarity, less populated clusters and low-cardinality representatives. By the converse, lower γ values correspond to a weaker intra-cluster similarity, huge clusters and high-cardinality representatives. By setting $\gamma = 1$ we choose the intersection as a cluster representative; by setting $\gamma = 0$ we choose the union.

3.3 Unclustered Objects

From the definition of d_J , the elements with empty intersection have distance 1. This means that objects having an empty intersection with each cluster representative are not inserted in any cluster. We refer to these unclustered objects as *trash*. This problem is mainly due to the fact that, within the domain \mathcal{U} equipped with d_J several objects can be equally distant from the other clusters. Consequently, any assignment of such objects to a cluster is not significant.

In section 5 we show that the cardinality of the trash is related to the clustering parameters (K and γ) and to the structure of dataset (Number of input objects, average cardinality of sets, number of distinct values). To overcome the problem of large trashes, various solutions can be adopted that act over these parameters.

- The initial cluster centers can be carefully chosen, in order to limit the size of the trash.
- We can iteratively reduce the value of the γ value, or augment the number of clusters.
- We can further apply the clustering technique to the trash, by choosing a random set of cluster representatives among the trash, and grouping the remaining elements re-iterating the algorithm. In such a way, we can avoid the effects of the high dissimilarity between the trash and the clusters previously generated.

4 Transactional K-Means

The main schema of the algorithm is shown in fig. 3. The schema is substantially similar to the traditional K -Means algorithm. However, The algorithm has two main phases. In the first phase, the algorithm computes $k + 1$ clusters. The tuples are assigned to the first k clusters according to the distance measure d_J . $(k + 1)$ -th cluster (the trash cluster) is created to contain objects that are not assigned to any of the first k clusters.

The second phase has the main objective to manage the trash cluster. The main idea in this phase is to recursively apply the algorithm with different parameters. As explained in the previous section, different choices can be done at this point. We adopted the solution to try to recursively split the trash cluster into two further clusters. The idea is that of maintaining the approach as stable as possible, by trying to split the trash cluster into a minimal number of clusters. Of course, the final result may contain clusters with a single element: elements substantially different can remain in the trash cluster until they are not chosen as cluster centers.

4.1 Implementation

An experimental version of the algorithm was implemented in C++. Such an implementation uses three main data structures: *itemset*, *cluster* and *kmeans*. An itemset is a collection of items, where an item is represented by a pair identifier-frequency. Itemsets are kept sorted according to their identifier, in order to speed-up the main operations, such as union, intersection and distance.

The adoption of boolean vectors to represent itemsets is not a suitable approach, unless we deal with small datasets defined on a small \mathcal{I} . In real-life datasets, this solution requires handling of huge data structures because of the presence of thousands of distinct categorical values. As an example, we consider a real-life dataset containing 5961 web sessions containing 57571 distinct web pages and 8799 web sites. The resulting boolean representation requires 40Mb to code the dataset at the "page" abstraction level and 6Mb at the "site" abstraction level. A comparison of two objects requires the inspection of the whole

Algorithm TrK-Means(D, k, γ);

Input : a dataset $D = \{x_1, \dots, x_N\}$ of transactions, the desired number k of clusters. A cluster representative threshold value γ .

Output : a partition $\mathcal{C} = \{C_1, \dots, C_{k+l}\}$ of D in $k+l$ clusters, where $l \geq 0$.

Method :

- Randomly choose x_{i_1}, \dots, x_{i_k} and set $m_j = x_{i_j}$ for $1 \leq j \leq k$.
 - Repeat
 - for each j , set $C_j = \{x_i | d_j(x_i, m_j) < d_j(x_i, m_l), 1 \leq l \leq k\}$;
 - set $C_{k+1} = \{x_i | \text{for each } j \ d_j(x_i, m_j) = 1\}$;
 - set $m_j = \text{rep}(C_j)$ for $1 \leq j \leq k$;
 - until m_j do not change;
 - recursively apply the algorithm to C_{k+1} , producing a partition of C_{k+1} in l clusters.
-

Fig. 3. The Transactional K-Means Algorithm

set of items. This is clearly a rather expensive operations, if we consider that the average length of a session in the dataset is 26.

A solution to this problem is to adopt efficient compression techniques of sparse matrices. However, we can use a more compact representation of objects, by means of lists implemented by variable-length vectors. In our example, categorical values are coded using 4 bytes, and the resulting dataset has 600Kb size using the complete range of values.

A *cluster* implements the maintenance of the cluster representative. At each iteration, as soon as an object is recognized to belong to the cluster, a list of all the items belonging to the elements of the cluster is updated, and their frequency is tuned accordingly. After the inspection of all the transactions, the representative is recomputed selecting all the frequent items. *kmeans* is the structure which contain all clusters. To save cluster assignments of objects it maintains a mapping structure, implemented by means of an hash table.

4.2 Complexity analysis

The complexity of the algorithm depends from the complexity of a single iteration. The assignment of object to the nearest cluster is $O(NKc(d_j))$, where N is the number of objects in the database, K the number of required clusters and $c(d_j)$ is the cost of the computation of the distance between two objects. Assuming that $c(d_j)$ is linear in the size of a transaction, the total cost of the cluster assignment phase is at most $O(NKM)$. The representative constructions is at most $O(KM)$, since, for each cluster we have to check the frequency of each item inside a cluster. Assuming to keep ordered all the cluster representatives, the total cost of an iteration is $O(NKM + KM \log M)$. The second addend of the sum is significant only if $\log M$ is of the same order of N . This hypothesis is hardly verified in real datasets.

As a result, the complexity of the iteration is, in the worst case, linear in the size of the dataset.

5 Experimental Results

The objective of the following experiments is to study the behavior of the algorithm. To this purpose,

- We first compare the performance of the algorithm with the performance of the standard K -means representation that uses a boolean-vector representation.
- Next, we analyze the scalability of the approach on large transactional datasets.
- We study how the parameters of the algorithm, i.e., the γ threshold and the number k of cluster influence the trash population. In order to do this, in these tests we do not further partition the trash cluster.

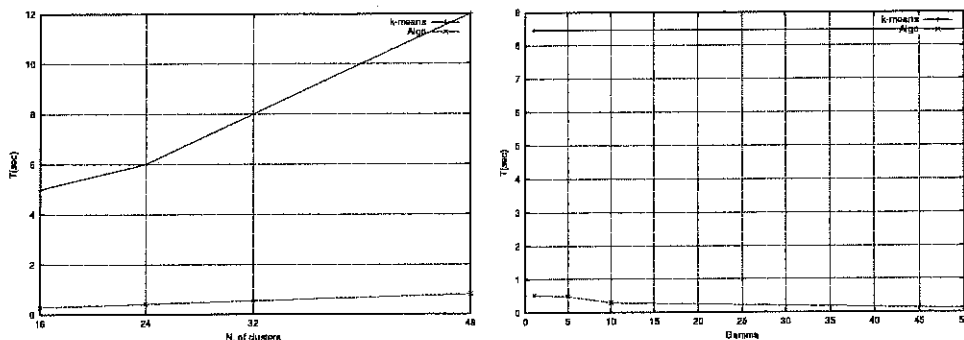


Fig. 4. Transactional Clustering vs. Standard K -Means: varying K and γ , $T = 30$.

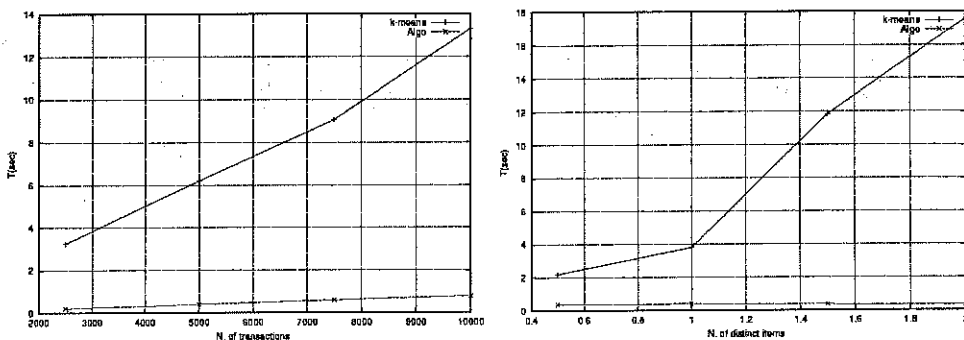


Fig. 5. Transactional Clustering vs. Standard K -Means: varying N and $|Z|$, $T = 30$.

- Finally, in order to test the efficacy of the approach, we analyze the results of the algorithm on a real dataset containing web usage logs.

5.1 Performance

In this phase we use the synthetic data generator available at [16], which allow us to create arbitrary sets of transactions. The parameters we shall consider are N , the total number of transactions, $|Z|$, the number of distinct items and T , the average length of transactions.

Experiments were made on a Linux system with a 400Mhz Intel Pentium II processor, 128Mb RAM and a 10000rpm hard disk. We chose not to load the entire dataset in main memory, since this option can be unpractical with very large datasets. Better, the tuples are read in blocks and stored in main memory at need. It is worth noticing, however, that the data structures presented in the above section, needed to represent the transactions, are substantially smaller than the bitmap representations used by standard K -Means, and typically fit into main memory producing significant performance improvements.

Comparison with K -Means. In order to compare the two algorithms we have implemented a K -Means version which works on a boolean data matrix, and uses the traditional euclidean distance and cluster representative. Figures 4 and 5 show the average iteration time of the two approaches. As expected, the overhead due to the boolean-vector representation is significantly large.

Performance on Large Datasets. The following graphics show the changes in total response time and average response time for iteration w.r.t. the clustering parameters K and γ and dataset properties N , T and $|Z|$.

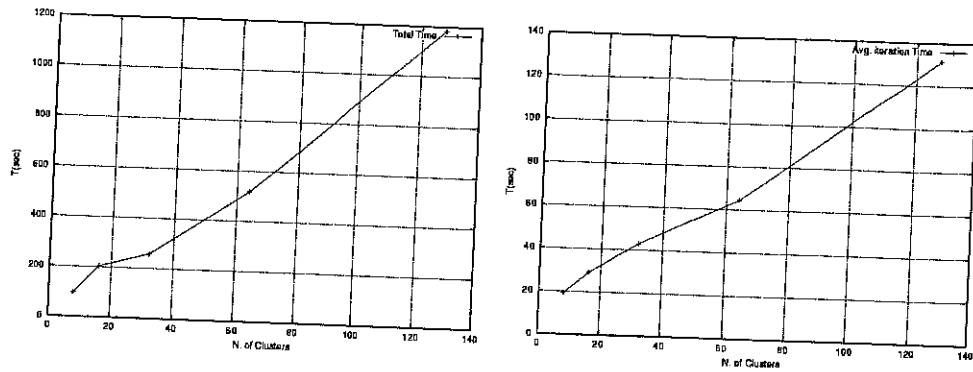


Fig. 6. Total Time and Average Time for Iteration: varying K .

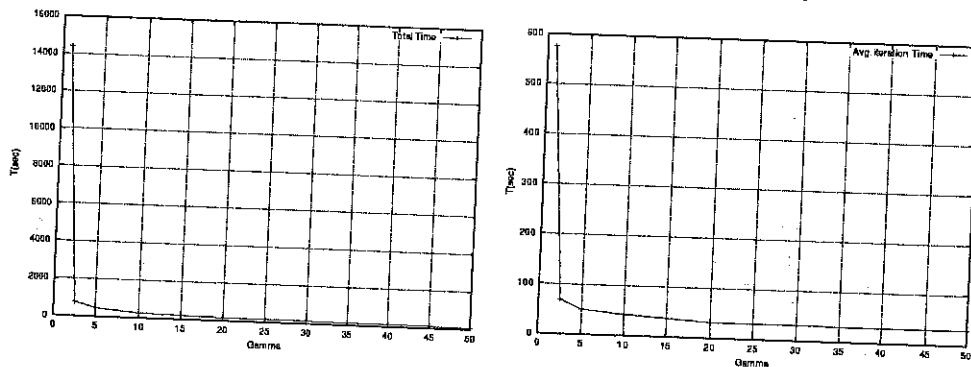


Fig. 7. Total Time and Average Time for Iteration: varying γ (in percentage).

Figures 6, 8 and 9 confirm the linear scalability of the algorithm w.r.t. the number of clusters, number of transaction and average cardinality of transactions increase. The execution time is inversely proportional to the value of the γ threshold (fig. 7): lower gamma values correspond to larger representatives and consequently to an increasing cost for computing the cluster assignment.

Trash Analysis. The Trash population is extremely significant w.r.t. the clustering results. Intuitively, a higher number of clusters corresponds to a smaller trash population. At the same time, the trash population is affected by the γ threshold value, that fixes the intra-cluster degree of similarity desired by the user. If γ is high, the clusters contains object with high similarity. Consequently, a large number of objects are likely to be contained within the trash cluster. Figure 10 shows the behavior of the trash population for varying values of K and γ .

The size of the trash population can be affected by the dataset characteristics, too. A greater number of distinct items and transactions (fig. 11) corresponds to a higher probability of having almost different transactions. On the other side (fig. 12), transactions with a high average size are more likely to be similar.

5.2 Application to Real Datasets

In order to prove the effectiveness of our algorithm we provide a case study concerning Web sessions clustering. Further experiments concerning document classification and clustering of socio-economic data in presence of null values is available at [6].

The main objective of the experiment is to study the behavior of a typical (anonymous) web user coming from a given community. To this purpose, the idea of using the web logs of a proxy server of a

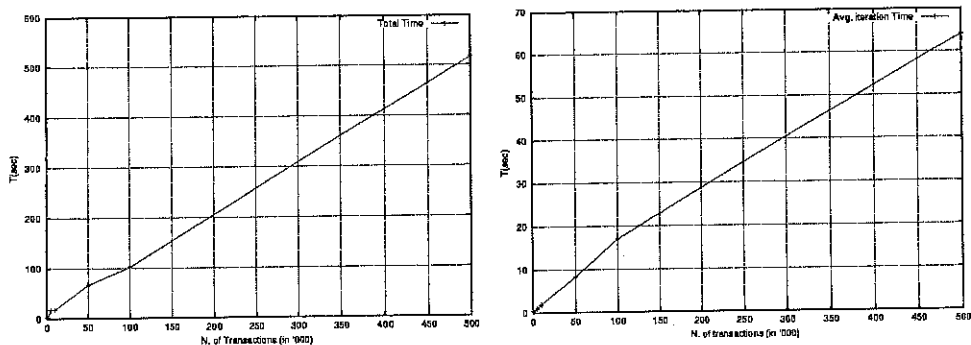


Fig. 8. Total Time and Average Time for Iteration: varying N (in thousands).

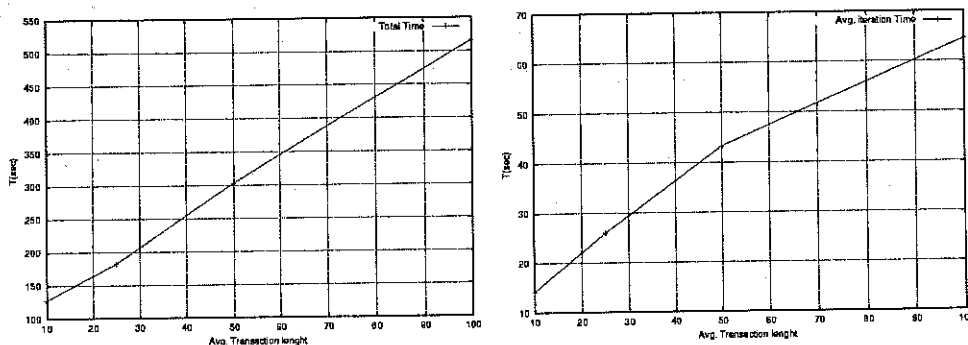


Fig. 9. Total Time and Average Time for Iteration: varying T .

given community gives us sufficient information about the browsing patterns of the the users belonging to that community.

We made our experiments with the data available from the web logs coming from the proxy server of the University of Pisa. A web log is a file in which each line contains a description of the access to a given web resource from a given user. A typical log row contains information about the IP of the user requesting the resource, the address of the resource requested, the size and status of the request and the time the request was made. We considered logs covering two weeks of browsing activity of the users of the University of Pisa.

In the preprocessing phase, we grouped the user accesses by client. Each web session corresponds to the sequence of pages that are requested by a given user in a reasonably low time interval [2]. In our experiment, the dataset contained 5961 sessions with average cardinality of 26 web accesses. We considered sessions containing information concerning web requests both at the "site" abstraction level and at the "resource" abstraction level. For both the levels we obtained good quality clusters with acceptable trash quantity. Figure 13 shows an example of the results obtained using the algorithm at site level with $\gamma = 0.05$ and $K = 32$. For this experiment, the initial cardinality of the trash cluster is 1335, and can be reduced to smaller values by recursively applying the algorithm. The resulting cluster representatives contain significant sites concerning related arguments. This peculiarity makes the cluster interpretation extremely simple. For instance, we can observe that cluster 1 groups sessions concerning linux resources, cluster 2 groups sessions concerning libraries and online document retrieval services, and cluster 3 groups sessions concerning on-line news.

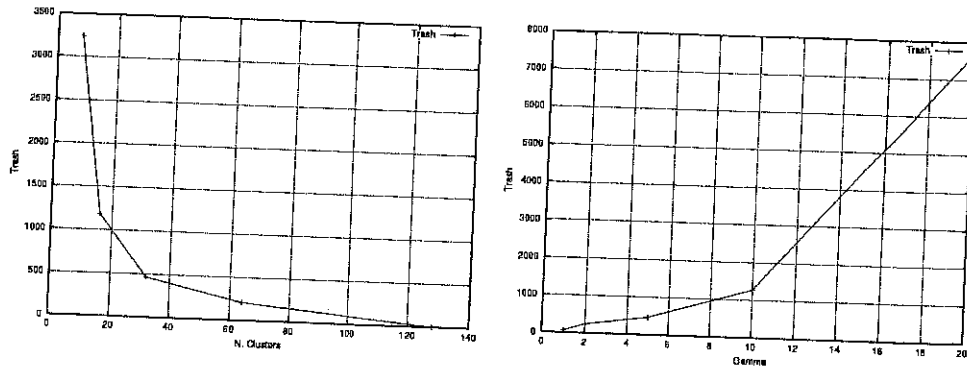


Fig. 10. Trash Population w.r.t. K and γ .

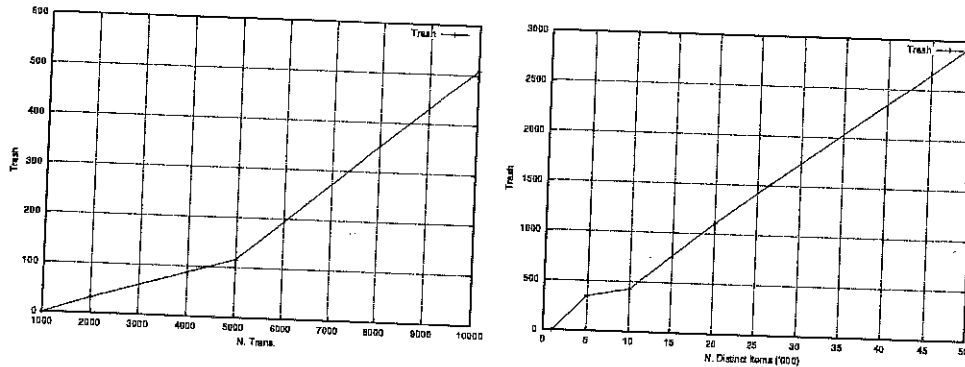


Fig. 11. Trash Population w.r.t. N and $|Z|$.

6 Conclusion and future work

The great advantage of the K -Means algorithm in data mining applications is its linear scalability. This makes it particularly suitable to deal with large datasets. However the approaches proposed to extend the use of K -Means to categorical attributes are suitable only for attributes with small domains, because of the large data structure needed to represent the inputs. The algorithm we have proposed crosses such limitations,

- replacing traditional vectorial representation of input objects with sets of non-fixed length;
- using a similarity measure capable to deal with sets of categorical objects
- using an efficiently computable frequency-based concept of cluster representative.

These extensions allow us to use our algorithm directly to cluster transactional data without the need to convert input data into boolean vectors. Formal and experimental results show that the algorithm maintains a linear scalability, while the overall cost of the algorithm is not affected by the number of distinct values in the attribute domains.

The effectiveness of the approach is also proved by the easy interpretability of the results, made viable by the definition of the cluster representative. Experimental results in this settings have shown that the approach can be successfully applied to most significant transactional domains, such as web-session clustering and document clustering.

As a future work, we plan to adapt the method shown in this paper to probabilistic approaches, i.e., to approaches that predict for each cluster a probability of membership of a transaction (in the style of [1]), rather than a raw membership. Such an approach, in our opinion, has the advantage of providing a better

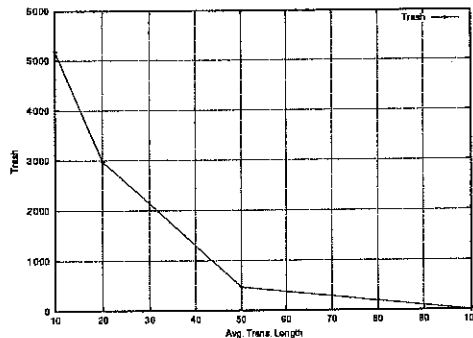


Fig. 12. Trash Population w.r.t. T .

www.linux-mandrake.com
wxstudio.linuxbox.com
Sunsite.cnlab-switch.ch
casema.linux.tucows.com
dada.linuxberg.com
www.kdevelop.org
www.newplanetsoftware.com
www.linuxberg.com
linuxpress.4mg.com
www.pluto.linux.it
linuxberg.concepts.nl
www.kde.org
www.its.caltech.edu

www.citeseer.com
www.informatik.uni-trier.de
search.yahoo.com
computer.org
www.google.com
citeseer.nj.nec.com
linwww.ira.uka.de
www.acm.org
www.yahoo.it

www.mondadori.com
www.repubblica.it
www.gazzetta.it
www.espressoedit.kataweb.it
www.kataweb.it

Fig. 13. Examples of Cluster representatives.

modeling of the dataset features, especially in domains in which different characteristics can be present in the same transactions.

References

1. I.V. Cadez, S. Gaffney, and P. Smyth. A General Probabilistic Framework for Clustering Individuals and Objects. In *Proceedings of the ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, USA, August 2000.
2. R. Cooley, B. Mobasher, and J. Srivastava. Grouping web page references into transactions for mining world wide web browsing patterns. In *Proc. of the 1997 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX-97)*, November 1997.
3. R. Dubes, and A.K. Jain. Clustering Methodologies in Exploratory Data Analysis. In *Advances in Computers*, Vol. 19, M. Yovits (ed.), Academic Press, pp. 113-228, 1980.
4. D. Fasulo. *An analysis of Recent Work on Clustering Algorithms*. Technical Report, April 1999. Available at <http://www.cs.washington.edu/homes/dfasulo>.
5. V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS - Clustering Categorical Data Using Summaries. In *Proceedings of the ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, USA, pp.73-83, August 2000.
6. C. Gozzi, F. Giannotti, and G. Manco. *Clustering Transactional Data*. Technical Report, CNUCE-CNR. Available at <http://www-kdd.cnuce.cnr.it>.
7. M. Grotschel, and Y. Wakabayashi. A cutting Plane Algorithm for a Clustering Problem. *Mathematical Programming* 45, pp. 59-96, 1989.

8. S. Guha, R. Rastogi, K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. In *Proceedings of the IEEE International Conference on Data Engineering*, Sidney, March 1999.
9. S. K. Gupta, K. Sambasiva Rao, and Vasudha Bhatnagar. K-Means Clustering Algorithm for Categorical Attributes. In *Proceedings of the Workshop on Data Warehousing and Knowledge Discovery (DaWaK-99)*, pp. 203-208, 1999.
10. E. Han, G. Karypis. Centroid-Based Document Classification: Analysis and Experimental Results. In *Proceedings of the European Conference on Principles of Knowledge Discovery in Databases (PKDD'00)*, pp.424-431, 2000.
11. Z. Huang. Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Mining and Knowledge Discovery* 2(3), pp.283-304, 1999.
12. J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceeding of the 5th Berkeley Symposium (vol. 1)*, pp. 281-297, 1967.
13. P. Michaud. Clustering techniques. *Future Generation Computer Systems*, 13, 1997.
14. P. Jaccard. The Distribution of the Flora of the Alpine Zone. *New Phytologist*, 11, pp.37-50, 1912.
15. H. Ralambondrainy. A Conceptual Version of the K-Means Algorithm. *Pattern Recognition Letters*, 16, pp.1147-1157, 1995.
16. R. Srikant. *Synthetic Data Generation Utilities*. IBM Almaden Research Center. Available at <http://www.almaden.ibm.com/cs/quest>.
17. M. Steinbach, G. Karypis, V. Kumar. A Comparison of Document Clustering Techniques. In *ACM-SIGKDD Workshop on Text Mining*, 2000.
18. A. Strehl, and J. Ghosh. A Scalable Approach to Balanced, High-dimensional Clustering of Market-baskets. In *Proceedings of High Performance Computing*, Bangalore, 2000.
19. A. Strehl, J. Ghosh, and R. Mooney. Impact of Similarity Measures on Web-page Clustering. In *Proceedings of AAAI workshop on AI for Web Search*, K. Bollacker (Ed), AAAI Press, pp.58-64, July 2000.
20. Ke Wang, Chu Xu, and Bing Liu. Clustering Transactions Using Large Items. In *International Conference on Information and Knowledge Management*, pp.483-490, 1999.
21. Oren Zamir, and Oren Etzioni. Web Document Clustering: A Feasibility Demonstration. In *International Conference on Research and Development in Information Retrieval*, pp.46-54, 1998.

