Clustering Transactional Data

Fosca Giannotti[1], Cristian Gozzi[1], and Giuseppe Manco[2]

[1] CNUCE-CNR, Pisa Research Area
Via Alfieri 1, 56010 Ghezzano (PI), Italy
F.Giannotti@cnuce.cnr.it
C.Gozzi@guest.cnuce.cnr.it
[2] ISI-CNR
Via Bucci 41c, 87036 Rende (CS), Italy
manco@isi.cs.cnr.it

Abstract. In this paper we present a partitioning method capable to manage transactions, namely tuples of variable size of categorical data. We adapt the standard definition of mathematical distance used in the K-Means algorithm to represent dissimilarity among transactions, and redefine the notion of cluster centroid. The cluster centroid is used as the representative of the common properties of cluster elements. We show that using our concept of cluster centroid together with Jaccard distance we obtain results that are comparable in quality with the most used transactional clustering approaches, but substantially improve their efficiency.

1 Introduction

The need to develop algorithms for clustering transactional data, i.e., tuples of variable size of categorical data, comes from many relevant applications, such as web data analysis. Log data typically contain collections of single accesses of web users to web resources, and the set of one user's accesses can be further collected into *sessions*. In such context there are some relevant difficulties that make the traditional approaches unsuitable, such as the enormous amount of transactions or distinct elements in a single transaction, that can be huge, and the inherent difficulty of classical algorithms to "semantically" cluster discrete-valued data. As an example, an important semantic feature in this context is the notion of cluster representative: namely, a transaction subsuming the characteristics of the transactions belonging to the cluster.

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. [4]. Here, the ROCK algorithm is presented, an agglomerative hierarchical method for clustering sets of categorical values. Hierarchical methods are often presented in the literature as the best quality clustering approaches, but they are limited because of their quadratic complexity over the number of object under consideration. Moreover, it is difficult to generate a representative providing an easy interpretation of the clusters population. An alternative to Hierarchical methods 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. However, these methods do not supply an adequate formalism for tuples of variable size containing categorical data.

The main idea of our approach is that of defining a new notion of cluster centroid, that represents 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. Our definition of cluster centroid is based on a data representation model which simplifies the one used in document clustering. In fact, we use compact representation of boolean vectors that states only presence and absence of items, while document clustering requires to store the frequencies of items (words).

In this paper we show that using our concept of cluster centroid associated with jaccard distance we obtain results having a quality comparable with other approaches used in this task, but we have better performances in term of execution time. Moreover, cluster representatives provide an immediate explanation of clusters features. A particular remark is due to the performance of our approach, which is relevant in case of real-time applications, e.g., clustering of search engine query results or web access sessions for personalization. In these cases, response time is a fundamental requirement of clustering algorithms, and performances of traditional methods is unacceptable.

The plan of the paper is as follows. Section 2 provides the formal foundations of the clustering algorithm that is shown in section 3. Finally, in subsection 3.1 we show formal and empirical results to prove that the algorithm is both efficient and effective on transactional data and in subsection 3.2 we briefly describe a real application on web log data.

2 Problem Statement

The most well-known partitional approaches to clustering data are the centroid-based methods, such as the K-Means algorithm [2]. In such approaches, 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 *centroid* (or *representative*) of the cluster.

Definition 1. *Given a set of objects $D = \{x_1, \ldots, x_n\}$, a centroid-based clustering problem is to find a partition $\mathcal{C} = \{C_1 \ldots C_k\}$, of D such that:*

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

□

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

The general schema of K-Means is parametric to the functions d and rep, that formalize respectively the concepts of distance measure and cluster centroid. 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 $\mathcal{S} = \{x_1, \ldots, x_m\} \subseteq \mathcal{U}$, the centroid of $\mathcal{S}$ is the element that minimizes the sum of the squared distances:*

$$rep(\mathcal{S}) = min_{v \in \mathcal{U}} \sum_{i=1}^{m} d^2(x_i, v)$$

□

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

Definition 3. *Given a set $\mathcal{I} = \{a_1 \ldots a_m\}$, where a_i is a categorical value (henceforth called* item*), The domain of transactional data is defined as $\mathcal{U} = \mathcal{P}owerset(\mathcal{I})$. A subset $I \subseteq \mathcal{I}$ is called an* itemset. □

Example 1. 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. ◁

2.1 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 2. Let us suppose $\mathcal{I} = \{a_1, \ldots, a_{10}\}$. 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:

I_1	$[1, 1, 0, 0, 0, 0, 0, 0, 0, 0]$
I_2	$[1, 1, 1, 0, 1, 1, 1, 0, 0, 0]$
I_3	$[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$

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 correponds to its absence. ◁

In principle, the above representation allows to directly apply commonly used distance definitions, such as Minkowsky or Mismatch count, and related optimal cluster centroids. 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 I_1 and I_2 and, by the converse, I_2 and I_3 are disjoint. Now, the above dissimilarity measures 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.

This problem has many common aspects with the problem of clustering documents. There, a document is coded as a term-vector, which contains the frequency of each significant term in the document. In this context, ad-hoc measures for these kind of data are used: *Jaccard Coefficient* [10] (henceforth called s_J), that computes the number of elements in common of two documents, and *Cosine similarity* [10] (henceforth called s_C), that measures the degree of orthogonality of two document vectors.

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

In the simplified hypothesis that term-vectors do not contain frequencies, but behave simply as boolean vectors (like in the case of web user sessions), a more intuitive but equivalent way of defining the Jaccard 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:

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

this measure capture our idea of similarity between objects, that is directly proportional to the number of common values, and inversely proportional to the number of different values for the same attribute.

2.2 Cluster Representative

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.

A problem with the traditional distance measures is that the computation of a cluster representative is computationally expensive. As a consequence, most approaches [1,9] approximate the cluster representative with the euclidean representative. However, such approaches suffer of some drawbacks:

– Huge cluster representatives cause poor performances, mainly because as soon as the clusters are populated, the cluster representatives are likely to become extremely huge.

– In transactional domains, the cluster representative does not represent a transaction.

On the other side, the use of the Jaccard distance for boolean vectors makes computing a cluster representative problematic as well. In fact, the problem of minimizing the expression $\sum_i d_J^2(x_i, c)$ given a set $\{x_1, \ldots, x_m\}$ cannot be solved in polynomial time if c is required to be a boolean vector [3]. In addition, an optimal cluster centroid is not necessarily unique.

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 1. *Given a set $\mathcal{S} = \{x_1, \ldots, x_m\}$, $rep(\mathcal{S}) \subseteq \bigcup_i x_i$* □

The above lemma states that the elements we need to consider in order to compute the representative are those contained in the union of the elements of the cluster. However, the cluster centroid can be different from the union. The idea of approximating the cluster centroid with the union has some drawbacks that make such a choice unpractical. First of all, the resulting representative can have a huge cardinality because of the heterogeneity of objects. 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 2. *Given a set $\mathcal{S} = \{x_1, \ldots, x_m\}$, $\bigcap_i x_i \subseteq rep(\mathcal{S})$* □

Also in this case, the intersection does not necessarily correspond to the cluster representative. Again, it can be unpractical to approximate the cluster representative with the intersection. With densely populated clusters, 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 3. *Given a set $\mathcal{S} = \{x_1 \ldots x_m\}$, $rep(\mathcal{S})$ is not empty.* □

A property of cluster representatives is that frequent items are very likely to belong to them. Such a property is not true in general: in cases where the most frequent items are contained in huge, non-homogeneous transactions, and there are also small transactions, the elements contained in small transactions are more likely to appear in the representative than the most frequent items. However, such a property is unlikely to hold in homogeneous groups of transactions, like in the case of transactions grouped according to Jaccard similarity. Hence, we can provide a greedy heuristic that, starting from the intersection, iteratively refines the approximation of the cluster representative by iteratively adding the most frequent items until the sum of the distances can be minimized. Figure 1

```
                    Algorithm $rep_H(\mathcal{S})$;

Input : A set of transactions $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$.
Output : A transaction $\mathbf{m}$ that minimizes $f(\mathbf{y}) = \sum_{\mathbf{x} \in \mathcal{S}} d^2(\mathbf{x}, \mathbf{y})$
Method :
    – sort $\bigcup \mathcal{S}$ by increasing frequency, obtaining the list $a_1, \ldots, a_m$ such that
      $freq(a_i, \mathcal{S}) > freq(a_{i+1}, \mathcal{S})$;
    – Let initially $\mathbf{m} = \bigcap_{\mathbf{x} \in \mathcal{S}} \mathbf{x}$;
    – while $f(\mathbf{m})$ decreases
        • add $a_h$ to $\mathbf{m}$, for increasing values of $h$;
```

Fig. 1. Greedy representative computation

shows the main schema of greedy procedure for representative computation. In principle, the procedure can produce a local minimum that does not necessarily correspond to the actual minimum. However,experimental results have shown that in most cases the heuristic computes the actual representative, and in each case it provides a suitable, compact approximation of the cluster representative.

Computing such an approximation can still be expensive, since we need to sort the items on the basis of their frequencies. We can define a further approximation wich avoids such computation, by introducing a user-defined threshold value γ over the frequency of the items appearing in any transaction of the cluster. Such a parameter should intuitively represents the degree of intra-cluster similarity 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 4. *Let $\mathcal{S} = \{x_1, \ldots, x_m\}$ be a set of transactions, and $\gamma \in [0, 1]$. The representative of $\mathcal{S}$ is defined as*

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

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

The approaches $rep_H(\mathcal{S})$ and $rep_\gamma(\mathcal{S})$ represent two viable alternatives: rep_γ represents an approximation that is, as we shall see, extremely efficient to compute. However, it is influenced by the value of γ. 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. On the other side, rep_H allows us to avoid specifying the threshold, at the cost of a less efficient computation.

2.3 Unclustered Objects

From the definition of d_J, the elements with empty intersection have distance 1. Using our distance 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. Using the mean vector as cluster centroid and jaccard or cosine distances there are not unclustered objects, because it is impossible to find objects having a null vector dot product with all cluster means. This is not always good, because unclustered objects can have a distance from the centroid very close to 1 and for this reason they must be considered outliers. Experimental results 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 centroids can be carefully chosen, in order to limit the size of the trash.
– We can iteratively reduce the value of 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.

3 Transactional K-Means

The main schema of the algorithm is shown in fig. 2. The algorithm has two main phases. In the first phase, it 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 try to recursively split the trash cluster into l further 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 centroids.

3.1 Experimental Results

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

```
Algorithm TrK-Means($D$,$k$,$\gamma$);

Input : a dataset $D = \{x_1, \ldots, x_N\}$ of transactions, the desired number $k$ of clusters.
    A cluster representative threshold value $\gamma$.
Output : a partition $\mathcal{C} = \{C_1, \ldots, C_{k+l}\}$ of $D$ in $k + l$ clusters, where $l \geq 0$.
Method :
    – Randomly choose $x_{i_1}, \ldots, x_{i_k}$ 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 $C_{k+1} = \{x_i |$ for each $j$ $d_J(x_i, m_j) = 1\}$;
        • set $m_j = rep(C_j)$ 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. 2. The Transactional K-Means Algorithm

- We first compare the performance and scalability of the two version of our algorithm (the one which uses the greedy procedure and the one which uses the γ threshold) with the performance of the traditional approaches (i.e., the approaches that adopt the mean vector as the cluster centroid and jaccard and cosine distance measures).
- Next, we compare the quality of results of the different approaches with real and synthetic datasets using some predefined quality measures. In this context, we study how the γ threshold and the number k of desired clusters influence the trash population.

In our experiments, we use both real and synthetic data. Synthetic data generation is tuned, among the others, according to [8, section 2.4.3] the number of transactions ($|\mathcal{D}|$), the average size of the transactions ($|T|$), and the number of different items (N).

We compared the performance of the approaches shown in this paper with the ROCK algorithm and two versions of the K-Means algorithm that use the mean vector as a representative, and respectively the Jaccard and Cosine similarity measures. Figure 3 contains two graphics that show respectively the comparison of total execution times of the various approaches, and the average iteration time of γ-based K-Means algorithm w.r.t. the number of transactions. Figure 4 shows the scalability of K-Means based approaches w.r.t. the number of distict items and the number of desired clusters. As expected, the overhead due to the mean vector centroid is significantly large. For datasets of little and medium size, ROCK is more efficient than K-Means with mean vector representative, but its execution time increases rapidly due to its quadratic complexity. Despite its high formal complexity, the greedy-based algoritm has better performances than the mean vector algorithms because of the lower cardinality of representatives, and than ROCK. It is worth noticing from fig. 3 and fig. 4 that K-Means based

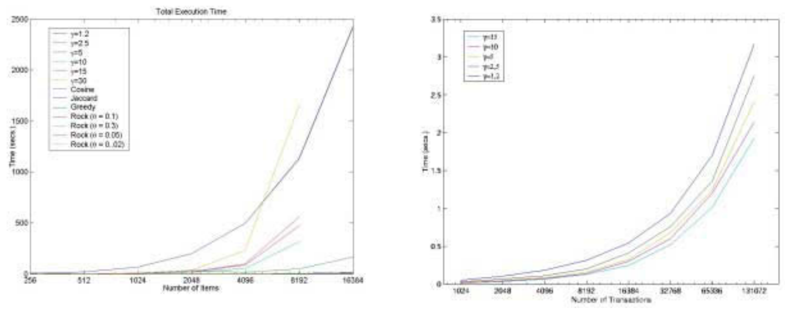

Fig. 3. Transactional clustering scalability varying N, $T = 30$, $D = 5K$

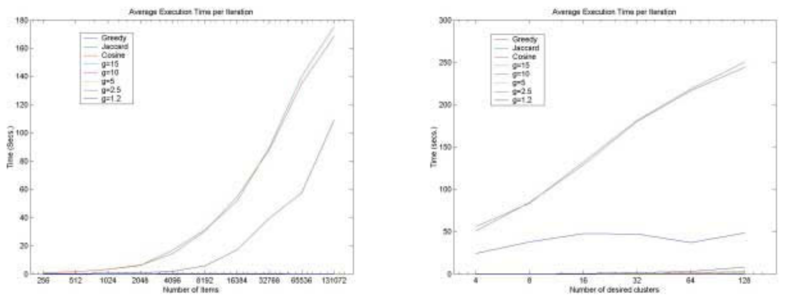

Fig. 4. Transactional clustering scalability varying D and K

algorithms scale, as expected, in linear time w.r.t. K, $|D|$ and $|N|$. However, the execution time of γ-based approach is inversely proportional to the value of the γ threshold: lower γ values correspond to larger representatives and consequently to an increasing cost for computing the cluster assignment.

To analyze quality of results we adopt two main approaches. The first approach is based on the consideration that the aim of a clustering algorithm is that of minimizing the intra-cluster dissimilarity [7]. The second approach, only suitable for synthetized data, adopts a further quality measure, namely the $\mathcal{F}$-measure [5].

According to the definition of clustering given in this paper, a natural quality measure associated to an instance of the k-Means approach that produces a partition $P = \{C_1, \ldots, C_{k+1}\}$ of the dataset D, where C_{k+1} is an additional cluster containing unclustered objects, is the following:

$$\mathcal{Q}_d = Avg_{i=\{1\ldots k\}} \frac{Avg_{\mathbf{x},\mathbf{y} \in C_i} d(\mathbf{x},\mathbf{y})}{Avg_{\mathbf{x},\mathbf{y} \in D} d(\mathbf{x},\mathbf{y})}$$

This quality measure computes the average of the average percentages of intra cluster distances respect to total distance between transactions. One drawback of the above measure is that large quantities of unclustered objects cause a better quality values: high γ values can cause a large trash quantity, but they cause also an higher intra-cluster similarity. Thereby, in order to evaluate the quality of $\mathcal{Q}_d$ measure one should consider also the number of unclustered objects.

When using synthesized data, we can adopt as a further quality measure the $\mathcal{F}$-measure [5]. We provide a class label for each transaction (where the number of different class labels, C, is a further parameter to be specified in the data generation process). A transaction $t = \{a_1, \ldots, a_n\}$ is assigned to a class by, first, randomly assigning a class frequency to each item belonging to the transaction, and, next, choosing the class c $(1 \leq c \leq C)$ that maximizes the formula

$$L(c) = \prod_{i=1}^{n}(1 + freq(a_i|c))$$

The $\mathcal{F}$-measure of a cluster i w.r.t. a class j is given by evaluating the *relevance* of the cluster:

$$\mathcal{F}(i,j) = \frac{2 \times p(i,j) \times r(i,j)}{p(i,j) + r(i,j)}$$

where $p(i,j)$ and $r(i,j)$ represent respectively *precision* and *recall* of i w.r.t. j. The total $\mathcal{F}$-measure is given by the weighted average of all values of the maximal $\mathcal{F}$-measure for every class. Higher values of the $\mathcal{F}$ measure represent higher quality clusters.

We compare the results quality of greedy-based and threshold-based algorithm with the ROCK algorithm and the other K-Means methods that adopt the mean vector with Jaccard and Cosine distance. We use three datasets as a testbench: the first two are synthetic datasets, and differ mainly in the number of distinct items $|D|$ and in the average transaction length $|T|$. As soon as $|T|$ and $|D|$ increase, transactions are less likely to be similar, and consequently the trash size is likely to increase as well. This phenomenon is even more evident in the third dataset, representing web user sessions, that tend to be extremely heterogenous. Table 1 resumes the quality results of the approaches. As we can see, the approaches show comparable results, even though threshold-based K-Means algorithm and ROCK are more sentitive to threshold values.

3.2 An Application: Web Sessions Clustering

In this section we describe a sample application of the algorithm: clustering of web sessions. 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 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

Table 1. Quality measures: Synth1=T100.D10k.N1k.C8, Synth2=T30.D1k.N1k.C8

Synth 1	*Greedy*	*Jaccard*	*Cosine*	γ= 10	γ= 5	γ= 2.5	*rock0.05*	*rock0.03*	*rock0.01*
$\mathcal{F}$	0.62439	0.70300	0.70561	0.72288	0.70402	0.69156	0.52927	0.83265	0.7384
$\mathcal{Q}_d$	0.92195	0.93612	0.93612	0.93472	0.93715	0.94389	0.9283	0.93961	0.94036
Trash	143	0	0	0	0	0	0	0	0

Synth 2	*Greedy*	*Jaccard*	*Cosine*	γ= 10	γ= 5	γ= 2.5	*rock0.05*	*rock0.03*	*rock0.01*
$\mathcal{F}$	0.68695	0.69645	0.64357	0.70808	0.66988	0.56565	0.68926	0.68232	0.45594
$\mathcal{Q}_d$	0.91893	0.91724	0.91726	0.91565	0.92732	0.93996	0.90508	0.90366	0.93002
Trash	87	0	0	0	0	0	0	0	0

Web data	*Greedy*	*Jaccard*	*Cosine*	γ= 10	γ= 5	γ= 2.5	*rock0.05*	*rock0.03*	*rock0.01*
$\mathcal{Q}_d$	0.78195	0.9489	0.9446	0.90361	0.9072	0.93411	0.9151	0.93012	0.95471
Trash	3717	0	0	2350	1981	1667	0	0	0

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 [6]. In our experiment, the dataset contained 5961 sessions with average cardinality of 26 web accesses. For this dataset we obtained good quality clusters with acceptable trash quantity. Figure 5 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 activating the second phase of 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 online news. On a total of 32 clusters, 20 cluster are directly understandable with a simple manual scan of the representative.

4 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 overcomes such limitations, using a similarity measure capable to deal with sets of categorical objects and using efficiently computable (frequency-based) concepts of cluster representatives.

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
liinwww.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. 5. Examples of Cluster representatives

These extensions make the algorithm proposed in this paper particularly suitable to on-line applications, i.e., applications that need an extremely fast computation of cluster assigments. Examples of such applications are clustering and reorganization of web search engines and analysis of web log accesses for on-line personalization. Formal and experimental results show the that the algorithm maintains a linear scalability, and contemporarily the overall cost of the algorithm is not affected by the number of distinct values in the attribute domain.

The effectiveness of the approach is also proved by the quality and easy interpretability of the results, made viable by the definition of the cluster representative. The main drawback of the approach, i.e., the presence of unclustered objects can be overcome by suitably adapting the technique as shown in section 2.3. However, in the perspective of using the algorithms for on-line applications, the problem of unclustered objects can be profitably ignored in favour of efficiency, provided that the quality of the clusters actually computed is acceptable.

References

1. I. Dhillon and D. Modha. Concept Decomposition for Large Sparse Data Using Clustering. *Machine Learning*, 42:143–175, 2001. 178
2. D. Fasulo. An analysis of Recent Work on Clustering Algorithms. Technical report, University of Washington, April 1999. Available at `http//www.cs.washington.edu/homes/dfasulo`. 176
3. M. Grötschel and Y. Wakabayashi. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45:59–96, 1989. 179
4. S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. In *15th International Conference on Data Engineering (ICDE '99)*, pages 512–521, Washington - Brussels - Tokyo, March 1999. IEEE. 175
5. J. Han and M. Kamber. *Data Mining Techniques*. Morgan Kaufman, 2001. 183, 184
6. R. Cooley B. Mobasher and J. Srivastava. Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns. In *Proceedings of the IEEE Knowledge and Data Engineering Exchange Workshop (KDEX-97)*, 1997. 185

7. L. Schulman. Clustering for Edge-Cost Minimization. In *Proceedings of the thirty-second annual acm symposium on Theory of computing*, pages 547–555, Portland, USA, May 2000. 183
8. R. Srikant. *Fast Algorithms for Mining Association Rules and Sequential Patterns*. PhD thesis, University of Wisconsin-Madison, 1996. 182
9. M. Steinbach, G. Karypis, and V. Kumar. A Comparison of Document Clustering Techniques. In *ACM-SIGKDD Workshop on Text Mining*, 2000. 178
10. A. Strehl, J. Ghosh, and R. Mooney. Impact of Similarity Measures on Web-page Clustering. In K. Bollacker, editor, *Proceedings of AAAI workshop on AI for Web Search*, pages 58–64. AAAI Press, July 2000. 178